Leverage Points for Systems Health Management of Autonomous Systems

Anupa R. Bajwa[1]
NASA Ames Research Center, Moffett Field, CA 94035

Systems Health Management (SHM) is one of three basic functionalities that constitute an autonomous capability of a system. The other two functionalities are Planning & Scheduling, and Task Execution. In an autonomous system, variable autonomy is often distinct from variable authority to sense, decide, and act. There are quantifiable Levels of Autonomy that can be achieved by tuning different portions of the Observe-Orient-Decide-Act loop to provide flexibility and control. This approach is tabulated for multiple domains such as spacecraft and aerial vehicles. Examining SHM through a Systems Thinking lens helps us understand its stocks and flows, loops, and delays. Systems thinking, and modeling, is a useful way to understand change and complexity of systems of many types. There are certain archetypes that underlie well-known autonomy architectures. And there often are leverage points – best places to intervene in a system – that can resolve or mitigate some fundamental challenges in the design and deployment of autonomous systems. I identify these levers and present the ones that have been successfully used in NASA missions.

I. Introduction

IN an autonomous system, tunable autonomy arises from combinations of varying degrees of capability, and authority, to sense, decide, and act. I propose, in Table 1, quantifiable Autonomy Levels that are achieved by tuning different portions of the Observe-Orient-Decide-Act loop.

These levels are based on the Autonomy Levels for Unmanned Systems (ALFUS) framework supported by the National Institute of Standards and Technology. ALFUS is a logical framework for characterizing autonomy for unmanned systems. It covers levels of autonomy, mission complexity, and environmental complexity. The Framework provides standard definitions, metrics, and process for the specification, evaluation, and development of the autonomous capabilities [6].

AL	Descriptor	Observe	Orient	Decide	Act
10	Full autonomy	Aware of status of fleet-wide assets	Autonomous fleet-wide awareness	Autonomous fleet coordination	Group accomplishes all goals
9	Swarm cognizance	Sensors and models to infer intent of other vehicles	Strategic group goals assigned, infers other vehicle intent	Distributed tactical group planning, individual goal determination	Group accomplishes strategic goals without assistance
8	Space environment knowledge	Proximity inference, reduced dependence on off-board data	Strategic group goals assigned	Coordinated, tactical team planning, individual task planning	Group accomplishes tactical goals with minimal operator assistance

[1] Co-Principal Investigator, Intelligent Systems Division.

7	Space environment sensing	Short-track awareness, limited inference plus off-board data	Tactical group goals assigned, estimates other vehicle trajectory	Individual task planning and execution	Accomplishes tactical goal with ground monitoring
6	Multi-vehicle cooperation	Range awareness, communication with other vehicles	Tactical group goals assigned, estimates other vehicle location	Coordinated trajectory planning and execution	Maintains close levels of separation
5	Multi-vehicle coordination	Local sensors to detect others, cloud communication	Tactical group plan assigned, compensates for failures	Onboard trajectory planning, optimize for current condition	Avoids collisions
4	Fault-adaptive vehicle	Deliberate state awareness, communication with ground	Tactical plan assigned, compensates for control failures	Onboard trajectory planning, self-manage resources	Maintains medium levels of separation
3	Scripted response to failures	Health/status sensors, history, and models	Real-time health diagnosis, adaptive inner-loop control	Evaluates status for required mission; safe mode if needed	Accomplishes tactical plan with assistance from operator
2	Full automation for nominal operations	Health/status sensors	Real-time health monitoring, off-board replanning	Executes preloaded sequence or upload new sequence	Executes original plan or new plan
1	Limited automation	Pre-loaded mission plan, spacecraft control, navigation sensing	Pre- and post-flight BIT	No deviation from pre-planned mission	Executes pre-planned mission sequence
0	Remotely-controlled	Traditional mission control	Downlinks data, responds to uplinked commands	No onboard decisions	Acts as commanded

Table 1 Autonomy Levels Mapped to Autonomy Functions for Space Missions

Future missions, such as for heliophysics, or lunar mobility/habitat will likely have a multi-craft architecture. Table 1 can help in the selection, and tuning, of portions of autonomous functionalities. However, a simple smorgasbord-like selection could create a system with an unsustainable or a lopsided structure, and hence misidentify its behavior, utility, and outcomes.

II. Identifying and Utilizing Leverage Points of a System

Systems Thinking offers a way to map a system so that all stakeholders can see a system, agree on its current state, understand how its variables are linked and how its delays propagate. It facilitates a systemic inquiry into the underlying structure that can impact the system's performance. Systems thinking maps are a visual representation of our mental model of the dynamics present in the system.

The core components of system maps are reinforcing and balancing causal loops. Reinforcing loops are a series of links that amplify each other in a self-reinforcing process. They may exhibit patterns of runaway growth or decline. Balancing loops are self-correcting processes. They may exhibit patterns of oscillation. Implicit in every balancing loop is a goal state that the system is trying to maintain.

System Archetypes are configurations of reinforcing and balancing loops that determine the system's behavior. There are intended and unintended consequences to every action. Archetypes help categorize and understand various interconnections, interplay, and relationships between elements in a system.

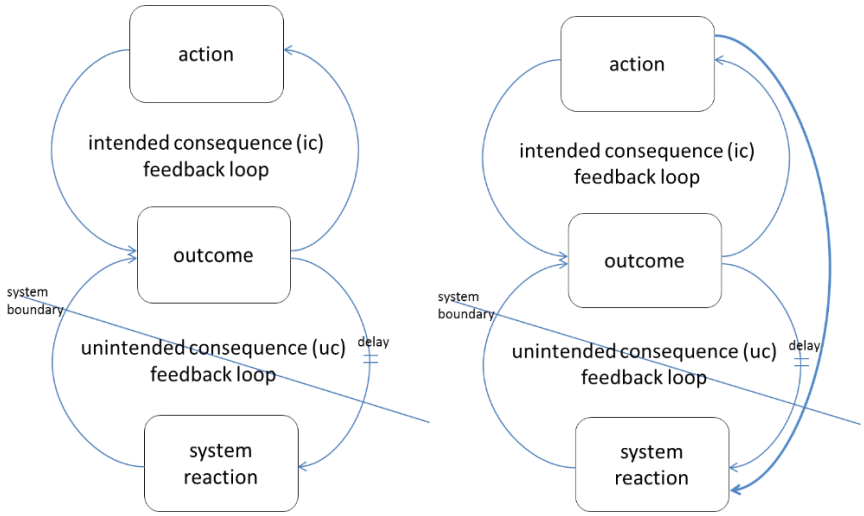

Fig. 1 Systems Thinking: the Problem Archetype (left) and the Solution Archetype (right).

Eric Wolstenholme [1] condenses system archetypes down to a more understandable core set of totally generic archetypes, consisting of the four ways of ordering a pair of reinforcing and balancing feedback loops. At the meta level, for every ''problem'' archetype he identifies a closed-loop ''solution'' archetype (Fig. 1).

Donella Meadows' influential article, Leverage Points: Places to Intervene in a System [2], was penned during a discussion of global trade treaties. The article identifies levers that can influence the outcome of a system. These are points of high leverage where a small change can have a large impact.

I map these leverage points to succinctly describe many aspects of defining, developing, and operating a spacecraft Fault Management (FM) system. The Top 12 Levers in Fault Management (in increasing order of effectiveness) are:

A) Select Parameters
B) Architect Buffers
C) Leverage Redundancy
D) Understand Lengths of Delays
E) Choose the Right Level of Response for Fault Management
F) Ride the Implementation Wave
G) Visualize Information Flow
H) Levy Requirements, Know Constraints
I) Refine the Wheel (but don't re-invent it)
J) Set the Goals of Fault Management
K) Acknowledge the Paradigm
L) Transcend the Paradigm

III. Leverage Points in the LADEE Fault Management System

The Lunar Atmosphere and Dust Environment Explorer (LADEE) spacecraft is a small orbiter, category II, enhanced Class D spacecraft built on a modular common bus architecture. It achieved its science goal to examine the structure and composition of the tenuous atmosphere of the Moon and to understand its dust distribution.

During the design phase LADEE's low-cost, single-string "common bus" backbone challenged the scope of its Fault Management system. The team relied on model-based tools to design command sequences, understand the consequence of an unexpected reboot, and test various strategies for ground controller intervention and override.

This section discusses how to use Systems Thinking's leverage points to design an FM system to enable a successful mission.

A. Select Parameters

Start with an exploration of how many state variables to monitor. These may be determined by sensor placement and coverage, and might be restricted by available compute cycles and memory. Then select which state variables to monitor, such as spacecraft body rates, propellant tank pressure, subsystem temperature, bus voltage.

Set thresholds and tune them. Set upper and lower bounds for at least two limit bands for each variable: red limits and yellow limits. These may be set by Subsystem Designers or Mission Operators, based on prior experience, or may be based on vendor recommendations. Setting sampling rates for onboard use (based on available storage) and for downlink (constrained by packet size).

B. Architect Buffers

In systems there are items that you can track, count, or measure at any point in time. These are referred to as levels or accumulations. These are affected by repeated behaviors or actions which result in inflows, and outflows.

Assess the FM data flow: will it be like a lake or like a river? Do consider using the stabilizing effect of buffers. Be sure to select right-sized buffering. If a buffer is too big, the system becomes inflexible. If too small, the system gets starved.

It is useful to identify and understand trade-offs between desired "ilities" of your architecture. Examples of such attributes, alphabetically arranged, are:

Accessibility; Availability
Capacity; Certifiability; Compatibility
Dependability
Effectiveness; Efficiency
Maintainability; Modifiability
Operability
Portability
Quality
Recoverability; Reliability; Resilience; Response time; Reusability; Robustness
Safety; Scalability; Security; Stability; Supportability
Testability; Transparency; Trustability
Usability
Versatility

C. Leverage Redundancy

Some spacecraft missions have the luxury of multiple redundant subsystems. A backup can be deployed if the primary subsystem, or a component, fails. For instance the fault protection on the Cassini mission [7] had to be robust so that no credible single point failure prevents attainment of the objectives, or results in a significantly degraded mission. Exemptions (Fig. 2) were granted for items whose failure probability was low due to the presence of large design margins. Adequate physical redundancy was provided with four reaction wheels for 3-axis control, three RTGs, and two main engines.

For tighter mass budgets, such as on LADEE, the engineer must leverage dissimilar functional redundancies. Identify alternate-use options, and identify all single point failures. Design the layout to match the redundancy. For instance, select whether to use a single, full-length harness or to use a split harness to connect the subsystems.

Estimate how many spares to create and store – for example the Lunar Reconnaissance Orbiter (LRO) [3] set included spare Transponder, Diplexers, Band Reject Filter, 6-dB Couplers, RF Transfer Switches.

SPF#	Cassini Single Point Failures (Exemption List)
1	Loss of a Radioisotope Thermoelectric Generator
2	Loss of 1 High Gain Antenna (HGA), or either Low Gain Antenna (LGA 1 , LGA 2) inside 1.5 AU
3	Leakage or bursting of a propulsion module tank (pressurant tank, main engine oxidizer tank, main engine fuel tank, thruster)
4	External leakage or bursting of propulsion module fluid or pressurant lines and fittings…
5	Structure (Spacecraft adapter, orbiter, or Probe)
6	Spacecraft separation band (retention / release)
7	Thermal blankets, surfaces, and shields (spacecraft and probe)
8	Spacecraft cabling short
9	Selected command and data errors
10	Main engine combustion chamber (catastrophic explosion)
11	Passive radio frequency equipment
12	Micrometeoroid shielding (inherent or specific)
13	Power interruption greater than 37 milliseconds
14-18	Probe adapter structures, Probe structure, spin-up and release mechanisms

Fig. 2 Exempted Single-Point Failures for Cassini.

D. Understand Lengths of Delays

All systems have delays in them. Delays are the "no feedback" zones of a system. To handle the delay of propagation of physical effects in a system use the "persistence" parameter of a state variable to determine whether it has crossed a pre-defined threshold limit. This helps deal with sensor noise as well as with the effect propagation time. Relate this to sampling rate, and trade against threshold limits. Use timid or bold limits based on risk tolerance.

Delays are accounted for by "wait" times, after power on, before using a component, or for a maneuver to complete (Fig. 3).

Event	Duration
Configure FM for LOI	15m
Thermal Conditioning	2h
Set Filter Tables for Maneuver	1m
Power ON IMU	1m
Configure Kalman Filter to use IMU rates	1m
Slew to Burn Orientation	15m
Power ON VDU	1m
Set Burn Duration	1m
Arm thrusters	1m
Verify Orientation, Arm Duration & Burn Orientation	15m
OUT OF COMMS BEHIND MOON	35m
Power OFF Transmitter (if necessary)	1m
Power ON Transmitter (if it was off)	1m
Re-Acquire Spacecraft, Establish Commanding	5m
Settling Burn	1m
LOI-1 Burn	4m
Rate Damping	1m
Rate Damping 2	1m
Preliminary Burn Performance Assessment	15m
Power OFF VDU	1m
Re-configure FM for Nominal Cruise	1m
Slew to Cruise Orientation	15m
Configure Kalman Filter for MEMS rates	1m
Power OFF IMU (if it's on)	1m
Set Filter Tables for Cruise	1m
Thermal Conditioning	1h
Establish Comms	5m
Downlink Maneuver Data	30m

Fig. 3 Representative Wait Times in a Lunar Mission

E. Choose the Right Level of Response for Fault Management

It is acceptable to not design automated responses to all failure conditions – sometimes "no response" is the right response for very low probability or very low severity failures. Most spacecraft, however, at least downlink simple telemetry to let the ground operator know the state of the spacecraft. This downlinked data contains measurements, status flags, or incrementing counters. The next level of automated response is to raise an event to notify operators.

A common strategy for spacecraft fault management is to execute an automated command to go to Safe Mode. As discussed in [5] spacecraft modes encapsulate many operational intentions. They define the feedback and actuators to control the spacecraft, and they provide a single command to switch between these mechanisms. In any particular mode (Fig. 4) the spacecraft can, in a pre-set way, automatically turn components on/off. Mode definitions, transition restrictions, and transition pathways often lead to lively debates between different design teams. For instance, there can be multiple options for the path to Safe Mode: with or without processor reboot.

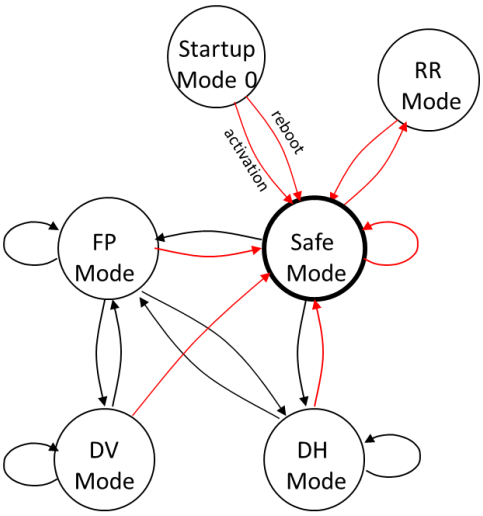

Fig. 4 Commanded (black) or Autonomous (red) Mode Transitions on LADEE

Further levels of corrective actions include sending a direct command from ground to recover from Safe Mode. Several missions have had to upload a new command sequence from the ground. If that does not resolve the issue, the next level of remediation involves uploading a new flight software load.

F. Ride the Implementation Wave

Leverage existing flight software (FSW) modules for FM functionality (Fig. 5). Leverage FSW for FM "failure injection" capability. Strive to "test like you fly" as best possible without risk of damaging the spacecraft by "injecting failures" into a software or a hardware simulator rather than the actual spacecraft. Address telemetry needs and command capabilities for Mission Operators, and data needs for Science Operations.

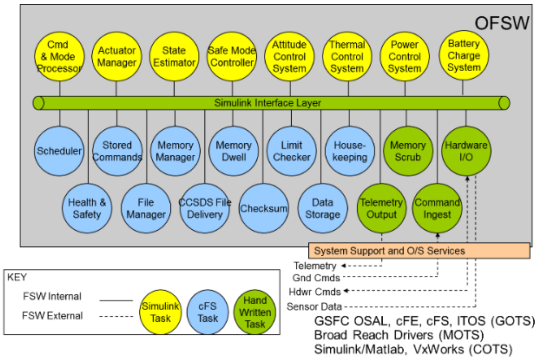

Fig. 5 LADEE Flight Software Architecture

G. Visualize Information Flow

Practice FM with Mission-Operations-in-the-loop (Fig. 6), to understand the right quantity, quality, and speed of information that needs to flow to the human operator.

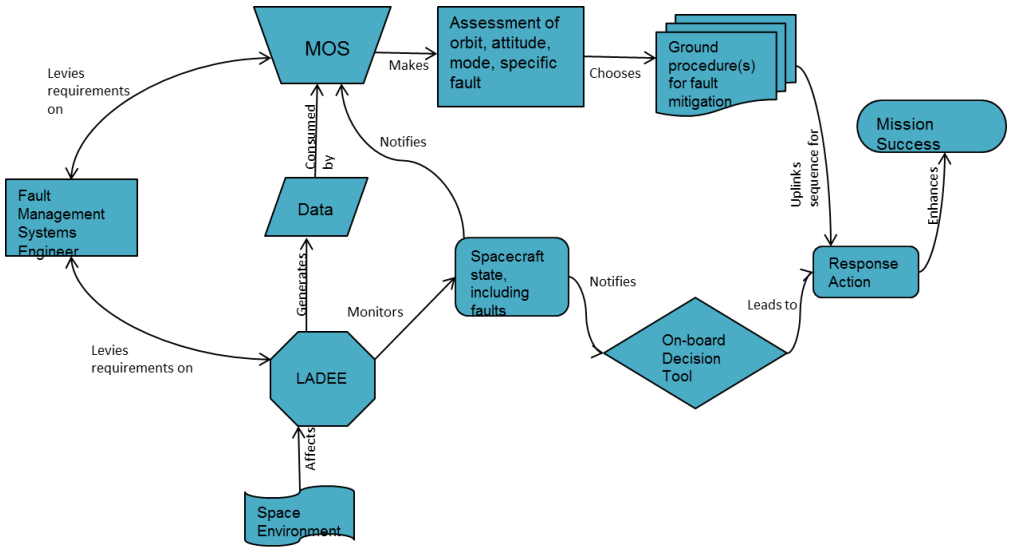

Fig. 6 Identifying Information Flow for LADEE Fault Management

H. Levy Requirements, Know Constraints

FM levies requirements on Flight Software, and vice versa – earlier the better! Additionally, FM levies requirements on itself – such as "do no harm". FM levies requirements on Mission Operations, and vice versa – iterate this along with first draft of Ops Concept.

FM often levies requirements on Science Operations. This may depend on the scope of FM under the constraints of CPU, memory, available spacecraft resources for continued operation such as power, propellant, pressurant, coolant.

An often underestimated constraint is the time available to design, implement, debug and verify FSW. Verification, alone, can take up to an hour per requirement on a medium-sized mission. This can be reduced through model-based development of software.

I. Refine the Wheel (but don't re-invent it)

Leverage lessons learned from similar NASA missions to design out the known issues and to be better informed about responding to spacecraft failures. For instance, the Kepler mission underwent multiple safing events, from false positives due to sensor noise (reference). Curiosity (Mars Science Laboratory) had unexpected computer resets, for which the underlying cause was a processor bug. Likewise, LRO had an operating system bug (two differing implementations of fmod function).

Capture this process knowledge to help future missions. For instance the Cassini fault protection had to be redesigned [8] from its original form [7]. One month after Cassini's launch when the propellant tanks were pressurized for the first time, the prime regulator was leaking at a rate significant enough to require a considerable change. This occurrence of new failure modes required design changes in thresholding algorithms.

Be aware that heritage can be a double-edged sword. Seek consensus on what to conserve and retain from a previous system.

J. Set the Goals of Fault Management

Implement an operational capability to detect and respond to conditions that interfere with nominal operations. Maintain a capability to continue to operate through critical events. That is, develop a capability to "fail operational".

Provide a common context for subsystem designers to verbalize "what can go wrong", especially at interfaces, and during mode- or phase-transitions [5]. Coordinate, design, and implement Fault Management by leveraging, where possible, existing Flight Software modules.

K. Acknowledge the Paradigm

A paradigm is a set of beliefs, cultivated over time, about how a mission or a project should be designed and implemented. A key factor is a project's risk posture based on mission risk classification.

Encourage Project commitment to the importance of early involvement of fault management. Start in the formulation phase (Phase A) to dampen the ubiquitous "workload bump" in the implementation phase (Phase D).

Strive for a buy-in from subsystems – this is hard! Many subsystem leads are certain that they have a perfect system. Their paradigm is to design the nominal system first. And they may see fault management capability simply as some additional software that can be folded in later.

L. Transcend the Paradigm

Earn the buy-in from subsystem leads through active listening. Start with a scenario-based discussion of potential failures in the subsystems, assess their likelihood and severity, and then flow these scenarios back into detailed requirements. The sooner a subsystem's failure handling requirements are implemented in software, the sooner the subsystem team gets to "try it out" in simulation – including a failure simulation.

Establish "certifiable trust" in system performance that cannot be anticipated from behavior of individual subsystems. This is especially valuable for responding to subtle failures during critical events [4].

As system complexity grows acknowledge that fault management may not scale elegantly. There may be a need to develop a flexible, autonomous control architecture with goal-driven adaptability to operate under all conditions.

IV. Conclusion

This is an initial exploration of a Systems Thinking perspective on Fault Management. The next effort needs to focus on system archetypes that help, or hinder, the development of effective fault management for NASA missions. Particularly, as more autonomy is introduced into the space and air vehicles of tomorrow, there is going to be a stronger need to understand old paradigms and their constraints. To infuse new ways of implementing bolder missions there needs to be a way of understanding system archetypes and how to overcome their limitations.

Acknowledgments

This research was conducted with formulation support from the Autonomous Systems Project in the Game Changing Development Program of NASA's Space Technology Mission Directorate.

References

[1] Wolstenholme E, "Towards the definition and use of a core set of archetypal structures in system dynamics," System Dynamics Review, Vol. 19, No. 1, Spring 2003: doi: 10.1002/sdr.259

[2] Meadows, D., "Leverage Points: Places to Intervene in a System," The Sustainability Institute, Hartland, VT, 1999.

[3] Tooley, C., Houghton, M., et al, "Lunar Reconnaissance Orbiter Mission and Spacecraft Design" Space Sciences Review (2010) 150: 23–62 doi: 10.1007/s11214-009-9624-4

[4] Cannon, H., Bajwa, A., Berg, P., Crocker, A. "LADEE Preparations for Contingency Operations for the Lunar Orbit Insertion Maneuver" *IEEE Aerospace Conference*, 07119199, Big Sky, MT, 2015

[5] Bajwa, A. and Berg, P., "Modeling Spacecraft Modes for Nominal and Off-nominal Operations", *AIAA InfoTech at Aerospace*, 977971, 2011

[6] NIST Special Publication 1011-II-1.0 AUTONOMY LEVELS FOR UNMANNED SYSTEMS (ALFUS) FRAMEWORK, URL: http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=823618 [retrieved December 2017]

[7] Slonski J., "System Fault Protection Design for the Cassini Spacecraft" doi: 10.1109/AERO.1996.495890

[8] Morgan, P., "Cassini Spacecraft's In-Flight Fault Protection Redesign for Unexpected Regulator Malfunction", 978-1-4244-3888-4/10